LLMalMorph：论使用大型语言模型生成恶意软件变体的可行性

摘要

Large Language Models (LLMs) have transformed software development and automated code generation. Motivated by these advancements, this paper explores the feasibility of LLMs in modifying malware source code to generate variants. We introduce LLMalMorph, a semi-automated framework that leverages semantical and syntactical code comprehension by LLMs to generate new malware variants. LLMalMorph extracts function level information from the malware source code and employs custom-engineered prompts coupled with strategically defined code transformations to guide the LLM in generating variants without resource-intensive fine-tuning. To evaluate LLMalMorph, we collected 10 diverse Windows malware samples of varying types, complexity and functionality and generated 618 variants. Our thorough experiments demonstrate that it is possible to reduce the detection rates of antivirus engines of these malware variants to some extent while preserving malware functionalities. In addition, despite not optimizing against any Machine Learning(ML)-based malware detectors, several variants also achieved notable attack success rates against an ML-based malware classifier. We also discuss the limitations of current LLM capabilities in generating malware variants from source code and assess where this emerging technology stands in the broader context of malware variant generation.

关键词：

Abstract

大型语言模型（LLMs）已经改变了软件开发和自动化代码生成。受这些进展的激励，本文探索了 LLMs 修改恶意软件源代码以生成变种的可行性。我们引入了 LLMalMorph，这是一个半自动化框架，它利用 LLMs 对代码的语义和语法理解来生成新的恶意软件变种。LLMalMorph 从恶意软件源代码中提取函数级信息，并采用定制设计的提示词结合策略性定义的代码转换，来指导 LLM 生成变种，而无需资源密集型的微调。为了评估 LLMalMorph，我们收集了 10 个不同类型、复杂度和功能的多样化 Windows 恶意软件样本，并生成了 618 个变种。我们详尽的实验表明，在保持恶意软件功能的同时，可以在一定程度上降低这些恶意软件变种对防病毒引擎的检测率。此外，尽管没有针对任何基于机器学习（ML）的恶意软件检测器进行优化，一些变种在对抗一个基于 ML 的恶意软件分类器时也取得了显著的攻击成功率。我们还讨论了当前 LLM 在从源代码生成恶意软件变种方面的能力局限性，并评估了这项新兴技术在更广泛的恶意软件变种生成背景下的现状。

Key Words:

目录

插图

表格

主要符号对照表

LLM	大语言模型的英文缩写

第 1 章 引言

恶意软件（Malware）继续随着技术的快速扩张而激增。到 2025 年，网络犯罪造成的损失预计将达到每年 10.5 万亿美元 [1]。每秒大约发生 19 万起新的恶意软件事件 [2]，而 2024 年勒索软件的平均赎金要求预计将达到每次攻击 273 万美元，较往年急剧上升 [3]。尽管经过数十年的研究和缓解努力，这些数字突显了恶意软件研究在当今不断演变的威胁环境中的紧迫重要性。

现代最具变革性的 AI 技术之一是大型语言模型（LLMs），它在自然语言处理（NLP）[4]–[6]、代码生成 [7]–[12] 以及代码编辑和重构等软件工程任务 [13]–[15] 中展现了非凡的能力。鉴于这些优势和进步，利用 LLMs 进行恶意软件源代码转换是自然的发展。最近一项针对全球行业 1800 名安全负责人的调查 [16] 发现，74% 的人正经历着显著的 AI 驱动的威胁，60% 的人感觉准备不足，无法抵御这些威胁。尽管当前的模型仅从文本生成功能完整的恶意软件存在显著局限性，但研究表明它们可以生成恶意行为者能够组装成可操作恶意软件的代码片段 [17]。LLM 能力的进步与恶意软件威胁的演变相结合，为对手使用这些模型创建新恶意软件并将现有代码库变异成更难以捉摸和更具破坏性的变种铺平了道路。尽管恶意软件源代码比二进制文件更难获取，但能够访问源代码的对手，例如恶意软件作者、泄露存储库的用户或修改开源恶意软件的人，仍然可以利用 LLMs 生成新的、更难检测的变种。这些模型使攻击者能够持续精进和扩展其武器库，从而大规模增加恶意活动的持久性和规避性。

1.1 先前的研究

先前的研究提出了各种创建恶意软件变种的方法 [17]–[23]。然而，这些方法在至少以下一个方面表现出局限性（如表1.1所示）(A) 大多数现有方法没有利用 LLMs 来转换恶意软件的源代码 [18]–[23]；(B) 大多数方法依赖迭代算法来生成恶意软件变种 [18]–[20], [22], [23]；(C) 使用 LLMs 进行变种生成的方法，直接从成功率低的提示词开始 [17]。此外，尚不清楚生成的恶意软件在规避广泛使用的防病毒引擎方面是否表现更优。鉴于目前的情况，我们的工作引入了一种与现有恶意软件变种生成方法截然不同的方法。与大多数先前主要依赖基于对抗性机器学习或基于搜索的方法的研究不同，我们的方法独特地利用 LLMs 在源代码级别进行操作。基本上，它从恶意软件源

代码开始，以高成功率和最少的手动工作生成变种。此外，我们的方法不需要迭代训练或基于搜索的优化，这使其与现有的恶意软件转换方法根本不同。因此，我们提出了一个尚未充分探索的新研究方向。

表 1.1　与先前研究的对比

方法	源代码	LLM 使用	无需训练或迭代	逃逸提升
Qiao, Yanchen, et al. [18]	否	否	否	是
Tarallo [23]	否	否	否	是
Malware Makeover [20]	否	否	否	是
MalGuise [22]	否	否	否	是
Ming, Jiang, et al. [21]	否	否	是	是
AMVG [19]	是	否	否	是
Botacin et al. [17]	否	是	是	否
LLMalMorph	是	是	是	是

1.2 问题描述

鉴于现有方法的局限性以及 LLMs（特别是代码生成方面）的最新进展，我们旨在回答以下问题——我们能否利用预训练 LLMs 的生成能力，无需额外微调，来开发一个半自动化且高效的框架，以生成保留功能语义的恶意软件变种，这些变种能够规避广泛使用的防病毒引擎和机器学习分类器？

1.3 我们的方法

在本文中，我们对上述问题给出了肯定的回答。我们设计、实现并评估了 LLMalMorph——一个专门用于生成用 C/C++ 编写的 Windows 恶意软件功能变种的框架。我们只专注于 Windows 恶意软件，因为它在消费者和企业环境中广泛使用，仍然是恶意软件最常针对的操作系统 [24], [25]。

LLMalMorph 结合了自动化代码转换和人工监督来生成恶意软件变种。该框架利用一个开源的 LLM，应用精心设计的转换策略和提示词工程，在保持结构和功能完整性的同时，高效地修改恶意软件组件。人机协同（human-in-the-loop）过程处理复杂转换和多文件恶意软件中的错误，允许进行调试和配置调整。这种半自动化方法也

使我们能够量化基于 LLM 从源代码生成恶意软件变种中的人力投入。

1.4 实验和分析

我们选择了 10 个不同复杂度的恶意软件样本，使用 6 种代码转换策略结合一个 LLM 生成了 618 个变种。我们使用主要依赖基于签名的检测和静态分析的引擎的 VirusTotal[1]和 Hybrid Analysis[2]评估了防病毒（AV）检测率，并测试了语义保留性。代码优化（Code Optimization）策略在两种工具上均持续实现了较低的检测率。平均而言，相对于每个样本的基准检测率，LLMalMorph 在 VirusTotal 上将简单样本的检测率降低了 31%，将三个更复杂样本的检测率降低了 10% 至 15%；在 Hybrid Analysis 上，与各自基准相比，四个样本的检测率降低了 8% 至 13%。除了 AV 工具外，我们还在一个基于机器学习（ML）的恶意软件分类器上评估了 LLMalMorph，并观察到在特定样本上，优化（Optimization）策略和安全（Security）策略取得了较高的攻击成功率（分别高达 89% 和 91%）。诸如优化、安全和 Windows API 修改等策略需要更多手动编辑，其中 Windows 和安全策略需要更高的调试投入。值得注意的是，四个样本中超过 66% 的规避型变种保留了其语义，这证明了 LLMalMorph 生成功能规避型恶意软件的能力。

1.5 贡献

总结而言，我们有以下贡献：

• 我们设计并实现了 LLMalMorph，一个实用的 Windows 恶意软件变种生成框架，它使用一个开源的 LLM 和基于提示词（prompt-based）的代码转换。

• 我们在 LLMalMorph 中设计了一个人机协同（human-in-the-loop）机制，以解决 LLM 在调试多文件恶意软件源代码和项目级配置方面的局限性。

• 我们进行了广泛的实验，从 10 个样本生成了 618 个恶意软件变种，并评估了它们在 VirusTotal 和 Hybrid Analysis 上的检测率和语义保留性，以及在一个机器学习分类器（ML Classifier）上的攻击成功率。

• 我们使用代码编辑工作量（code editing workload）比较了不同代码转换策略的有效性，并讨论了 LLM 所犯错误的类型。

[1]https://www.VirusTotal.com/gui/home

[2]https://hybrid-analysis.com/

1.6 开源

LLMalMorph 框架及其所有相关组件可在 Github[3]找到。

[3]https://github.com/AJAkil/LLMalMorph

第 2 章 背景

本节描述与恶意软件（malware）、其检测系统以及大型语言模型（LLMs）相关的各种预备知识。

2.1 恶意软件和检测手段

恶意软件（Malware）指的是对手或攻击者用来在用户不知情的情况下，未经授权访问数字设备以破坏或窃取敏感信息的恶意程序 [26]。它是一个统称（umbrella term），用于描述广泛的威胁，包括木马（Trojans）、后门（backdoors）、病毒（viruses）、勒索软件（ransomware）、间谍软件（spyware）和僵尸程序（bots）[27]，针对多种操作系统，如 Windows、macOS、Linux 和 Android，以及各种文件格式，如可移植可执行文件（Portable Executable, PE）、MachO、ELF、APK 和 PDF [28]。在入侵系统后，恶意软件可以执行各种恶意活动，例如渗透网络、加密数据以勒索赎金或降低系统性能。

检测引擎和工具采用各种方法和工具来检测恶意软件。它们可以大致分为静态（static）、动态（dynamic）和混合（hybrid）方法 [28], [29]。静态检测在不执行恶意软件的情况下对其进行分析，依赖于诸如 PE 头信息（PE header information）、可读字符串（readable strings）和字节序列（byte sequences）等特征 [28]。动态检测涉及在受控环境（例如沙箱，sandboxes）中执行恶意软件，以监视运行时行为，如注册表修改（registry modifications）、进程创建（process creation）和网络活动（network activity）[28], [29]。混合检测结合了静态和动态特征，使用诸如操作码（opcodes）、对系统的 API 调用（API calls to the system）和控制流图（control flow graphs, CFGs）等数据 [28]。此外，基于启发式的检测（heuristic-based detection）使用启发式规则（heuristic rules）静态分析代码并动态分析行为，以确定恶意性 [30]。

结论

参考文献

攻读学位期间发表论文与研究成果清单

致谢